- ORL is used to set (1) some bits

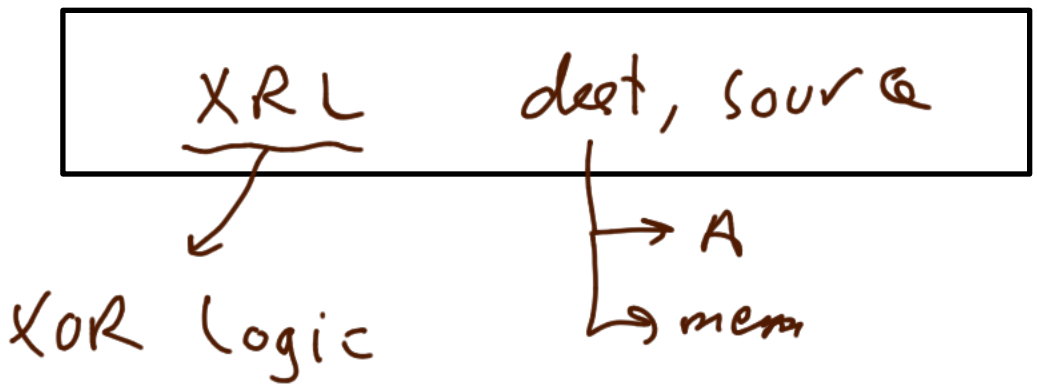
set bits 0, 5, 7 of A

```
ORL   A, #10100001B
```

- Is used to convert from unpacked BCD to ASCII

```
MOV   A, #04H
ORL   A, #30H
```

```
      0000 0100
ORL   0011 0000
      ---------
      0011 0100
        3    4
```



a	b	XRL
0	0	0
0	1	1
1	0	1
1	1	0

```
MOV   A, # 24H
XRL   A, # 56H
```

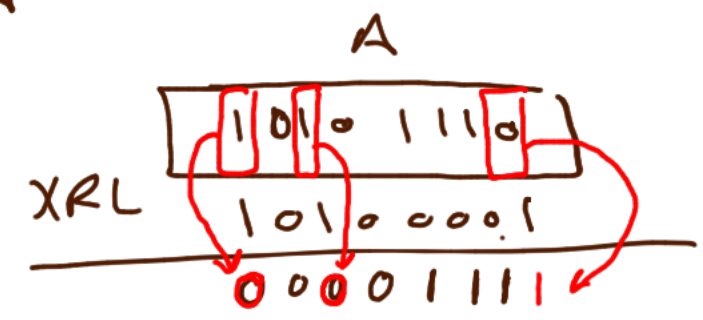
```
  0010 0100
XRL 0101 0110
-------------
  0111 0010
```

A = 72H

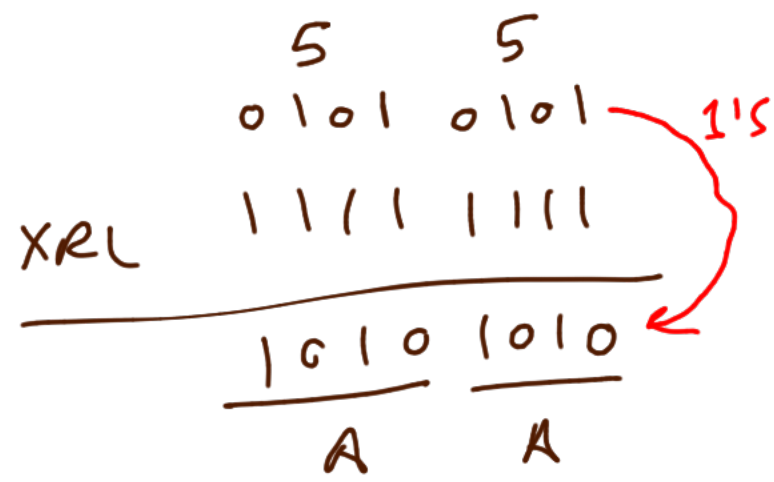
_ is used to complement (1's) some bits

complement (toggle) bits 0, 5, 7 of A

```
XRL   A, # 10100001B
```



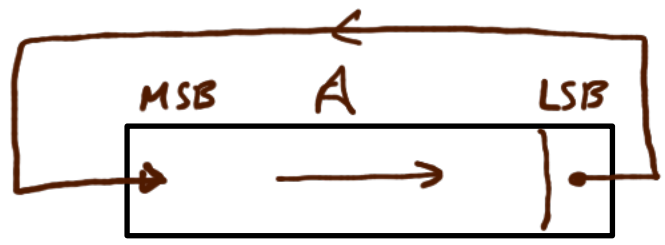
```
MOV   A, # 55H
XRL   A, # 0FFH
CPL   A
```



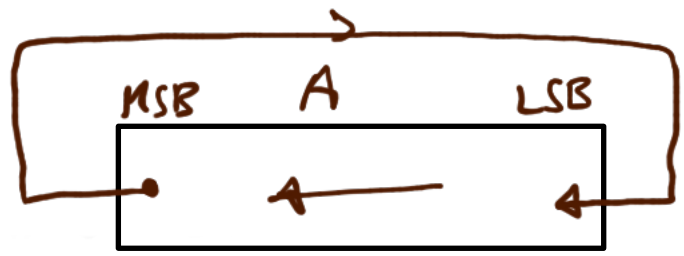
```
MOV   A, R2
XRL   A, R2   ; A = 00 H
```

Rotate (only with A)

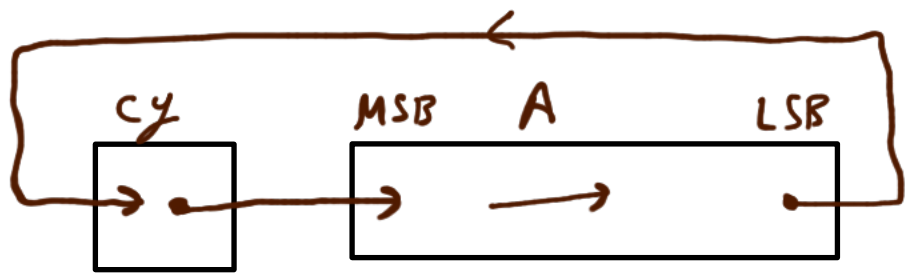
① `RR A` ; rotate right



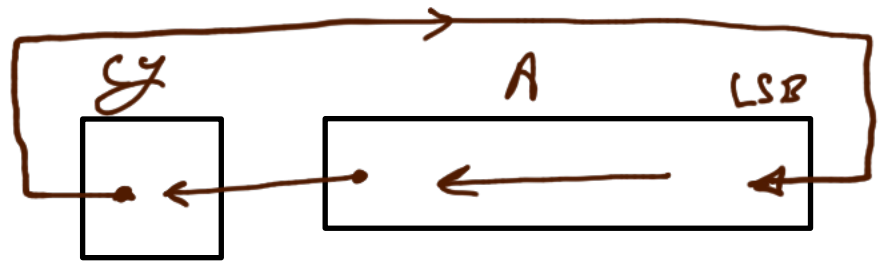
② `RL A` ; rotate left



③ `RRC A` ; rotate right with carry



④ `RLC A` ; rotate left with carry

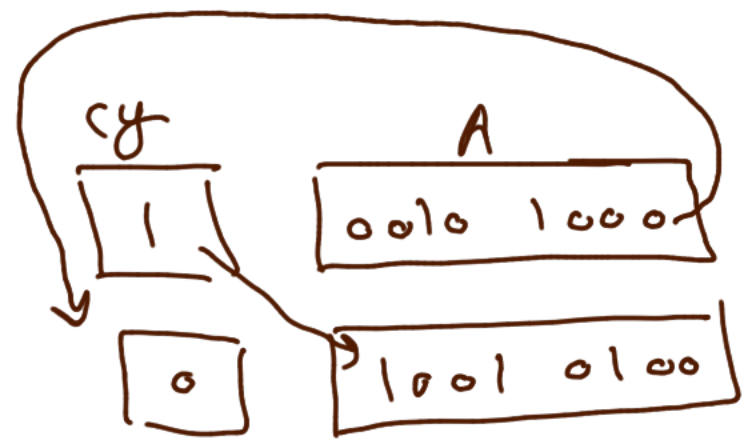


```
MOV   A, #  45H
RL    A     ; A = 9AH
RL    A     ; A = 15H
RL    A     ; A = 2AH
RL    A     ; A = 54H
SWAP  A     ; exchange low nibble of A
            with high nibble of A
```

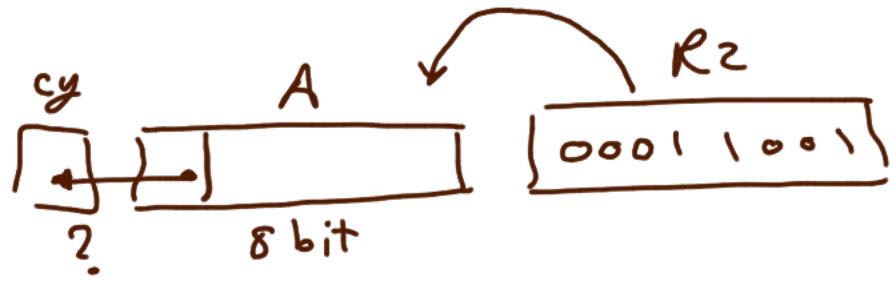
0100 0101 → 1000 1010 = 9AH → 0001 0101 = 15H → 0010 1010 = 2AH → 0101 0100 = 54H

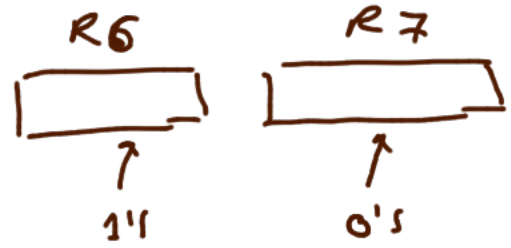
```
MOV   A, # 23H
SWAP  A    ; A = 32H
```

```
SETB  C
MOV   A, # 28H
RRC   A    ; A = 94H
           CY = 0
```



write a program to count number of 0's and 1's in R2.






```
        ORG   0000H
        MOV   R7, #0
        MOV   R6, #0
        MOV   R3, #8
        MOV   A, R2
Back:   RLC   A
        JNC   ZEROS
        INC   R6          = 1
        SJMP  NEXT
ZEROS:  INC   R7          = 0
NEXT:   DJNZ  R3, Back
        SJMP  $
        END
```